Lecture 14: Libraries and Encapsulation

Last time:
1. Parameter passing
2. Libraries
3. Public vs. private

Today:
1. Project #3 due 3/05
2. Exam #1 coming 3/07
3. APIs, comments and documentation
Project #3 Is Assigned!

- The assignment is on the CMSC 131 web-site (click “Projects” link).
- It is due Monday, 3/05 at 11 pm
- The project is open
- Start now!
  - Read entire assignment from beginning to end before starting to code
  - Check out assignment now from CVS
  - Follow the instructions exactly, as much of grading is automated
Exam #1 Is 3/05 in Discussion Section

- Take the test in your own section
- Test will cover material since beginning of semester
- Test will be closed book, closed neighbor
- Study!
Study When!?!  

- Project and test are very close to one another  
- This is part of college  
- What to do?  
  - Start project immediately  
  - Aim to finish project by Friday  
  - Start studying for exam on Saturday even if project not yet finished  
- You can do it!!
More on Project

- Project: Photo editing.
  - You are in charge of writing the PhotoTools Class
  - We provide the Driver, which will rely on your PhotoTools class
  - We provide a “library,” cmisc131PhotoLibrary, with two classes that you will need: (1) Photograph; (2) Pixel.

- Some new concepts
  - Collection of picture files (stored as .jpg)
  - Java API and Javadoc
  - Packages, Classes, Methods, Objects:
    - editing.PhotoTools.copy(p);
    - photo.PhotoSystem.begin();
    - p.getPixel(int x, int y);
    - r.getBlue();

- You must set up static methods inside of the PhotoTools class that can be called by external methods inside the “editing” package (or with an “import editing” statement):
  - PhotoTools.copy(p);
  - PhotoTools.makeBlackAndWhite(p);
  - PhotoTools.striped(p);
  - PhotoTools.weirdCombo(p);
How Do I begin?

By looking at the API documentation:

Other Project Tips

- **KISS** (Keep It Simple, S****d)
  - Read the documentation to identify the methods you need
  - Write very simple programs invoking the methods
  - Make sure you understand what the methods do!

- **Example:**
  - Photograph \( p = \textbf{new} \) Photograph(width, height);
  - Figure out how to determine width & height values
  - Understand when/how to use:
    - \( p\.\text{getWidth}() \)
    - \( p\.\text{getHeight}() \)
What is the API for the Cat example?

Class Cat

Constructor Summary
- Cat(String nameProvided)
  Create a new cat.
- Cat(Cat newCat)
  Create a copy of a cat

Method summary
- void setName(String newName)
  Set name of Cat
- void beEatenByADog()
  Removes a life from cat
- boolean isAlive()
  Tells whether cat has at least one life
- String toString()
  Returns string representation of cat (name and lives)
- String tellUsAboutCats()
  Provides a general description of cats

We will learn about the copy constructor next.
Copy Constructors

- Constructors can take arguments that are objects from the same class
  ```cpp
  Student (Student s) { ... }
  ```
- This is useful for defining *copy constructors*, e.g.
  ```cpp
  Student (Student s) {
    name = s.name;
    id = s.id;
    tokenLevel = s.tokenLevel;
  }
  ```
  Constructor creates a new object with same data as input
- So?
Example: Making a similar student

- Sometimes you want to create a similar student
  - Same name
  - Different id number
- Date class contains method
  ```java
  public void setID (int idIn) {...}
  ```
- Will the following create a new student with a new id?
  ```java
  Student henry = new Student ("Henry", 123456789);
  Student anotherHenry = henry;
  anotherHendry.id = 135792468;
  ```
- No!
  - `henry, anotherHenry` are aliases
  - Modifying `anotherHenry` also modifies `henry`
Using Copy Constructor

- To solve this problem, use copy constructor
  
  ```
  Student henry = new Student ("Henry", 123456789);
  Student anotherHenry = new Student (henry);
  anotherHenry.setID (135792468);
  ```

- **henry** now has id number 123456789
- **anotherHenry** has id number 135792468
Comments

- Explanations you add into your code
- Three forms in Java
  - /*
    Put text here
  */
  - // Put text here to end of line
  - /**
    Put text here
  */
- Last form is special comment for javadoc (utility for generating documentation from comments)
Comments (cont.)

- Some programmers hate them …
- … but they are essential for code understanding (it is called “code” for a reason)
- Beginning with Project #3, you will be graded on comments as well as indentation
- Comment:
  - Every variable (what is variable for?)
  - Every method (what does method do?)
  - Every class (what is class for? who wrote it?)